

## گزارش تمرین اول

برنامه نویسی میکروکنترلر جهت راه اندازی و کنترل انواع موتور الکتریکی

### سیستم‌های نهفته بی‌درنگ

اعضای گروه:

غزل کلهر، مبینا شاه‌بنده، امید بداقی، دیار محمدی

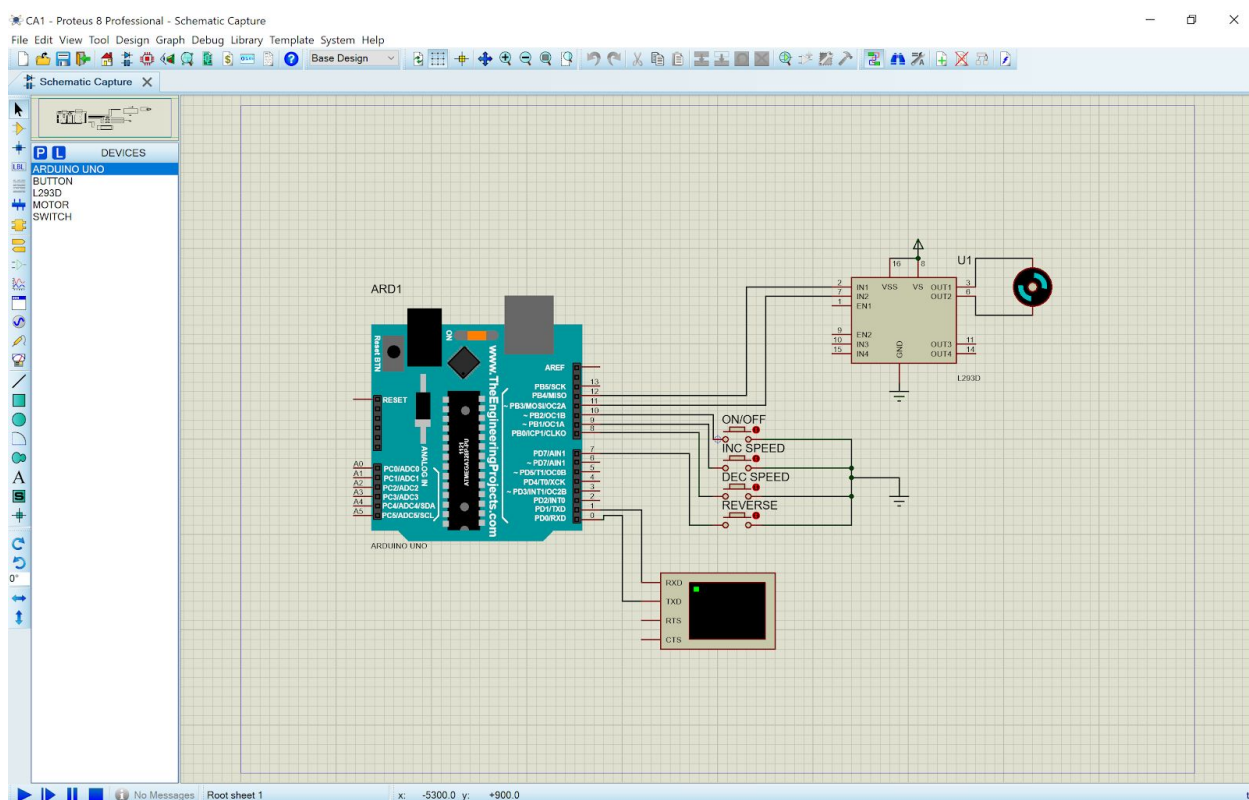
بهار ۱۴۰۰

## فهرست مطالب

2	بخش اول: کار با موتور DC
10	بخش دوم: کار با موتور Stepper
16	طراحی مفهومی مدار Stepper
17	شبیه کد مدار Stepper
18	پاسخ سوال 1
18	پاسخ سوال 2
20	پاسخ سوال 3
21	پاسخ سوال 4

## بخش اول: کار با موتور DC

در این قسمت هدف کار با موتور DC و اتصال آن به برد Arduino UNO است. این موتور به وسیله 4 دکمه، 4 عمل قطع یا ادامه حرکت، افزایش سرعت حرکت، کاهش سرعت حرکت و تغییر جهت حرکت را انجام می‌دهد. برای اتصال این موتور به برد Arduino UNO، از یک درایور L293D استفاده شده است. دو خروجی برد Arduino UNO به دو ورودی درایور و دو خروجی درایور به دو سر موتور متصل شده‌اند. دکمه‌ها از نوع BUTTON انتخاب شده‌اند. با هر بار فشار دادن دکمه‌ها، عمل متناظر توسط موتور صورت می‌گیرد. از یک virtual terminal برای مشاهده خروجی‌های چاپ شده توسط برد Arduino UNO استفاده شده است. شمای کلی مدار به صورت زیر است:



شمای کلی مدار موتور DC

برای برنامه‌ریزی برد Arduino UNO، در IDE کدی نوشته و خروجی hex کامپایل شده آن را به عنوان Program file در Proteus انتخاب می‌کنیم. حال بخش‌های مختلف کد را بررسی می‌کنیم:

در ابتدا ورودی های متناظر برد Arduino UNO و متغیرهای لازم (سرعت موتور، جهت چرخش، روشن و خاموش بودن موتور، شمارنده PWM) را تعریف می کنیم.

```
#define DC_MOTOR_PIN1 12
#define DC_MOTOR_PIN2 11
#define ON_OFF_BUTTON 10
#define INC_SPEED_BUTTON 9
#define DEC_SPEED_BUTTON 8
#define REVERSE_BUTTON 7
#define MAX_SPEED 255
#define MIN_SPEED 0
int motorSpeed;
int cnt;
bool motorOn;
bool clockwise;
```

سپس تابع setup که ابتدای شبیه سازی اجرا می شود را به صورتی می نویسیم که در ابتدای کار، موتور روشن، سرعت آن حداکثر، جهت چرخش ساعتگرد و شمارنده برابر با 0 باشد. همچنین برای نوشتن خروجی در ترمینال از Serial.begin استفاده می کنیم. عدد 9600 نشان دهنده سرعت نوشتن در خروجی (9600 بیت بر ثانیه) است که مقدار پیش فرض برای Arduino است. دو پین 11 و 12 برد Arduino UNO را به عنوان ورودی های دو سر موتور DC، و پین های 7 تا 10 را به عنوان خروجی های دکمه ها تعیین می کنیم.

```
void setup() {  
    motorSpeed = MAX_SPEED;  
    motorOn = true;  
    clockwise = true;  
    cnt = 0;  
    pinMode (DC_MOTOR_PIN1, OUTPUT);  
    pinMode (DC_MOTOR_PIN2, OUTPUT);  
    pinMode (ON_OFF_BUTTON, INPUT_PULLUP);  
    pinMode (INC_SPEED_BUTTON, INPUT_PULLUP);  
    pinMode (DEC_SPEED_BUTTON, INPUT_PULLUP);  
    pinMode (REVERSE_BUTTON, INPUT_PULLUP);  
    Serial.begin(9600);  
    Serial.println("DC Motor simulation");  
}
```

حال تابع loop که همواره در حال اجرا است را به صورتی می‌نویسیم که در آن با فشردن دکمه‌ها، عملیات متناظر در موتور DC رخ دهد. در ابتدای این تابع مقدار شمارنده PWM را یک واحد افزایش می‌دهیم. این شمارنده همواره در حال تغییر در بازه 0 تا 255 (حداکثر سرعت) است.

با فشردن اولین دکمه (دکمه‌ای که به ورودی 10 مورد وصل شده است) باید موتور از حرکت بایستد یا اگر متوقف شده بود به حرکت ادامه دهد. در حین فشردن دکمه نباید این عمل تکرار شود؛ به همین علت از while استفاده شده است تا با هر بار فشردن دکمه یک عملیات توقف یا ادامه حرکت صورت گیرد. این while در بخش‌های بعدی نیز آمده است که علت استفاده از آن مشابه همین قسمت است.

```
void loop() {  
  cnt = (cnt + 1) % MAX_SPEED;  
  
  if (digitalRead(ON_OFF_BUTTON) == LOW) {  
    Serial.println("on off");  
    while (digitalRead(ON_OFF_BUTTON) == LOW);  
    motorOn = !motorOn;  
    Serial.println(motorOn);  
  }  
  ...  
}
```

حال برای افزایش و کاهش سرعت حرکت موتور، در هر بار فشردن دکمه متناظر، سرعت را 10 واحد زیاد و یا کم می‌کنیم. حداقل سرعتی که می‌تواند تعیین شود برابر با 0 و حداکثر آن 255 است.

```
void loop() {  
    ...  
  
    if (digitalRead(INC_SPEED_BUTTON) == LOW) {  
        Serial.println("inc speed");  
        while (digitalRead(INC_SPEED_BUTTON) == LOW);  
        motorSpeed = (motorSpeed < MAX_SPEED - 10) ?  
            (motorSpeed + 10) : MAX_SPEED;  
        Serial.println(motorSpeed);  
    }  
  
    if (digitalRead(DEC_SPEED_BUTTON) == LOW) {  
        Serial.println("dec speed");  
        while (digitalRead(DEC_SPEED_BUTTON) == LOW);  
        motorSpeed = (motorSpeed > MIN_SPEED + 10) ?  
            (motorSpeed - 10) : MIN_SPEED;  
        Serial.println(motorSpeed);  
    }  
    ...  
}
```

برای تغییر جهت حرکت موتور، متغیر clockwise را در هر بار فشردن دکمه برعکس می‌کنیم. در انتهای کد، در حالت clockwise روی ورودی 1 موتور HIGH یا همان مقدار 1 و روی ورودی 2 موتور LOW یا همان مقدار 0 نوشته می‌شود و حالت counter-clockwise نیز برعکس آن است.

```
void loop() {  
  
    ...  
  
    if (digitalRead(REVERSE_BUTTON) == LOW) {  
        Serial.println("reverse");  
        while (digitalRead(REVERSE_BUTTON) == LOW);  
        clockwise = !clockwise;  
        Serial.println(clockwise);  
    }  
    ...  
}
```

در انتها در صورت روشن بودن موتور و کمتر بودن شمارنده PWM از سرعت تعیین شده، مقادیر HIGH و LOW (یک و صفر) با توجه به جهت حرکت موتور (که در بالا شرح داده شد) روی پین های خروجی برد که ورودی های موتور DC هستند، نوشته می شود. در صورت توقف حرکت موتور یا کمتر نبودن شمارنده، روی هر دو پین خروجی LOW نوشته می شود. در صورتی که ورودی بالایی موتور 1 و ورودی پایینی آن 0 باشد، موتور در جهت ساعتگرد و در صورتی که ورودی بالایی 0 و ورودی پایینی 1 باشد، موتور در جهت پادساعتگرد حرکت می کند. در صورتی که هر دو ورودی یکسان باشند موتور حرکت نمی کند.

همچنین در حالتی که شمارنده PWM بزرگتر مساوی سرعت تعیین شده باشد، موتور حرکت نخواهد کرد. بنابراین مجدداً بر روی دو پین خروجی مقدار LOW نوشته می شود.



```

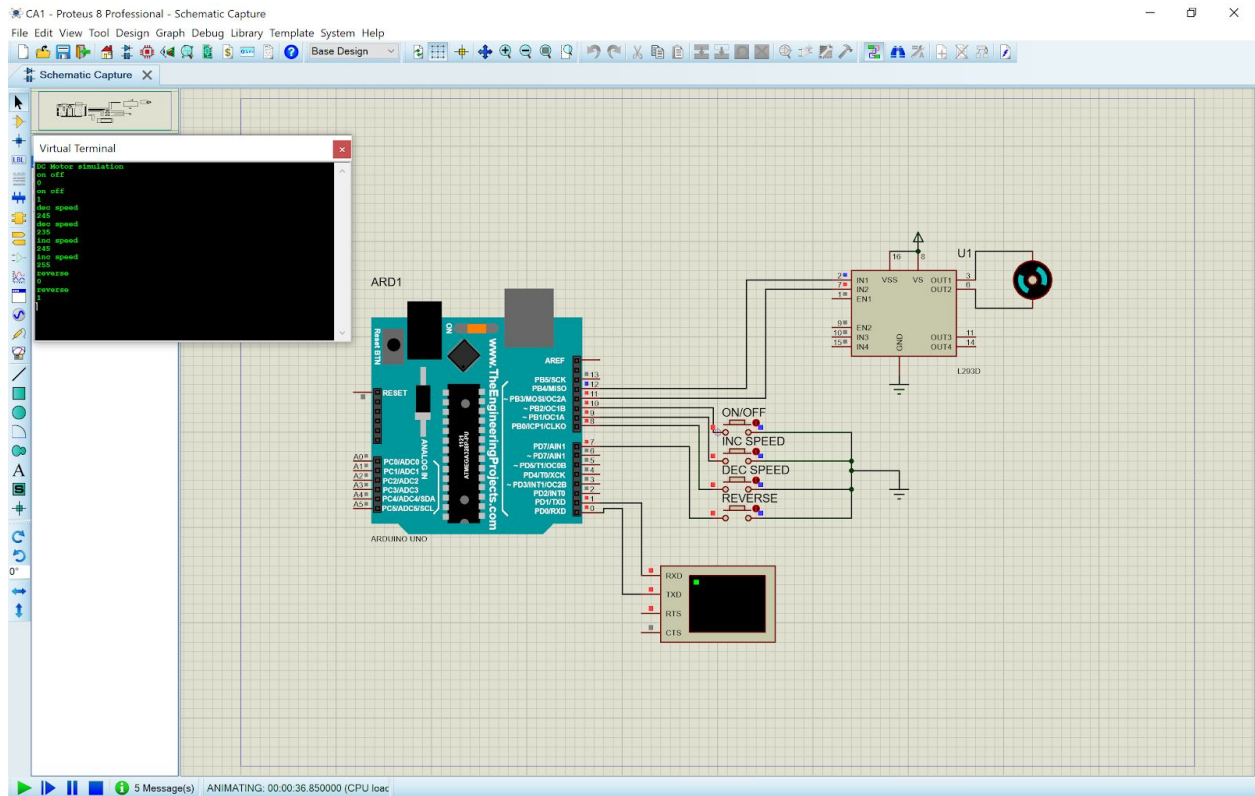
void loop() {

...

    if (motorOn && (cnt < motorSpeed)) {
        if (clockwise){
            digitalWrite(DC_MOTOR_PIN1, HIGH);
            digitalWrite(DC_MOTOR_PIN2, LOW);
        } else {
            digitalWrite(DC_MOTOR_PIN1, LOW);
            digitalWrite(DC_MOTOR_PIN2, HIGH);
        }
    } else {
        digitalWrite(DC_MOTOR_PIN1, LOW);
        digitalWrite(DC_MOTOR_PIN2, LOW);
    }
}
}

```

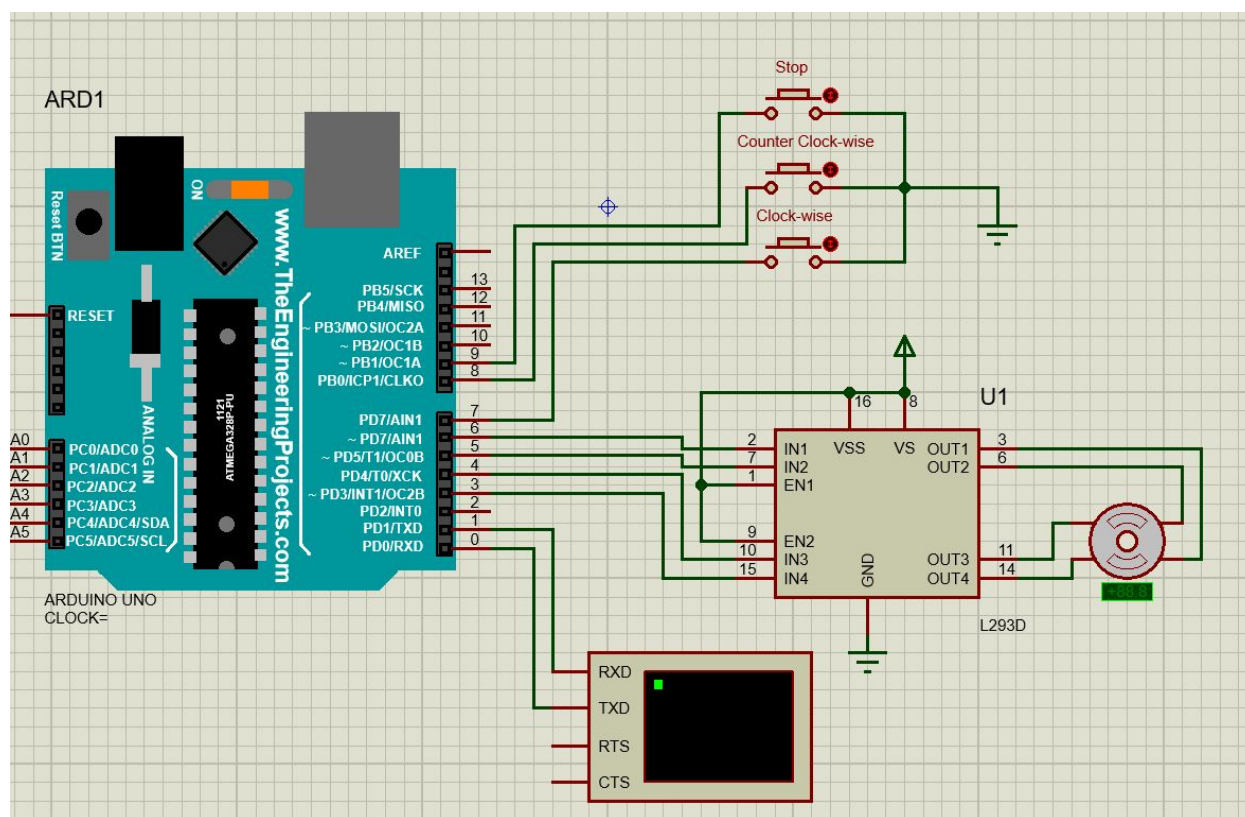
در ادامه نتیجه شبیه‌سازی را مشاهده می‌کنید که در آن ابتدا دکمه توقف/ادامه فشرده شده که موتور از حرکت ایستاده و مقدار 0 چاپ می‌شود و سپس این دکمه مجدداً فشرده شده و موتور به حرکت خود ادامه داده و مقدار 1 چاپ می‌شود. سپس دکمه کاهش سرعت موتور دو بار فشرده شده که منجر به کاهش 20 واحدی سرعت شده است (سرعت از 255 به 235 تغییر کرده است). پس از آن دکمه افزایش سرعت نیز دو بار فشرده شده که منجر به افزایش 20 واحدی سرعت شده است (سرعت از 235 به 255 تغییر کرده است). در انتها دکمه تغییر جهت چرخش موتور فشرده شده که ابتدا از ساعتگرد به پادساعتگرد تغییر یافته و با فشار دوباره دکمه، از پادساعتگرد به ساعتگرد تغییر یافته است. عدد 1 چاپ شده در ترمینال، نشان دهنده چرخش ساعتگرد و عدد 0 نشان دهنده چرخش پادساعتگرد است.



### نتیجه شبیه سازی مدار موتور DC

## بخش دوم: کار با موتور Stepper

در این قسمت هدف کار با موتور Stepper و اتصال آن به برد Arduino UNO است. این موتور به وسیله 3 دکمه، 3 عمل قطع چرخش، چرخش ساعتگرد و چرخش پاد ساعتگرد را انجام می دهد. برای اتصال این موتور به برد Arduino UNO، از یک درایور L293D استفاده شده است. خروجی های 3 تا 6 از Arduino به ورودی های درایور متصل شده اند و از درایور چهار پورت به موتور وصل شده است. دکمه ها از نوع BUTTON انتخاب شده اند. با هر بار فشار دادن دکمه ها، عمل متناظر توسط موتور صورت می گیرد. از یک virtual terminal برای مشاهده خروجی های چاپ شده توسط برد Arduino UNO استفاده شده است. شمای کلی مدار به صورت زیر است:



شمای کلی مدار موتور Stepper

مانند قسمت قبل خروجی hex کد Arduino به عنوان Program File به Arduino Uno داده شد. حال به بررسی کد می پردازیم:

### کتابخانه‌ی `stepper.h`:

برای استفاده از این کتابخانه، ابتدا باید آن را `include` کنیم. `Stepper(steps, pin1, pin2, pin3, pin4)` برای ساخت یک `instance` از موتور `Stepper` در آردوینو استفاده می‌شود. `Steps` تعداد گام‌ها در یک چرخش موتور را مشخص می‌کند. برای مثال، اگر هر بار قرار باشد ۱۰ درجه موتور بچرخد، تعداد قدم‌ها برابر با  $360/10 = 36$  خواهد بود. همچنین ۴ ورودی بعدی، ۴ پین وصل شده به موتور را مشخص می‌کنند. خروجی این تابع، `instance` ای خواهد بود که در ادامه از آن استفاده خواهیم کرد.

`setSpeed(int n)` سرعت موتور بر حسب RPM را مشخص می‌کند (باعث حرکت موتور نمی‌شود).

`step(int n)` باعث می‌شود موتور به تعداد `step` مشخص شده، با سرعتی که در تابع `setSpeed` مشخص شد، بچرخد.

### کد `Arduino`:

طبق توضیحات قسمت قبل، کتابخانه مربوطه `include` شده‌است. درجه چرخش و اندازه گام و پین‌های ورودی و کنترلی (حرکت ساعتگرد، پادساعتگرد و توقف) مشخص شده‌اند. همچنین متغیر `stopStepping` برای مشخص کردن اینکه موتور در حال حرکت است یا خیر و `motorDirection` برای مشخص کردن ساعتگرد یا پادساعتگرد بودن حرکت تعریف شده‌اند. در انتها با نام `groupStepper` را تولید می‌کنیم.

```
#include <Stepper.h>

#define degree 10
const int stepsNum = 360/degree;
#define STEPPER_MOTOR_PIN1 6
#define STEPPER_MOTOR_PIN2 5
#define STEPPER_MOTOR_PIN3 4
#define STEPPER_MOTOR_PIN4 3
#define CLOCKWISE_BUTTON 7
#define COUNTER_CLOCKWISE_BUTTON 8
#define STOP_BUTTON 9
bool stopStepping;
int motorDirection;

Stepper groupStepper(stepsNum, STEPPER_MOTOR_PIN1,
STEPPER_MOTOR_PIN2, STEPPER_MOTOR_PIN3, STEPPER_MOTOR_PIN4);
```

در قسمت `setup`، مقداردهی اولیه مقادیر را انجام می‌دهیم. ۴ پین استفاده‌شده برای `stepper` باید از نوع `OUTPUT` باشند. (چرا که خروجی `بورد Arduino` هستند) همچنین پین‌های کنترلی از نوع `INPUT_PULLUP` تعریف شده‌اند (چرا که ورودی `بورد Arduino` هستند).

سرعت موتور و سایر موارد مشابه قسمت قبل تکرار می‌شود. دقت‌شود حالت موتور در انتها، فعال است و جهت حرکت آن، ساعتگرد است.

```

void setup() {
  pinMode (STEPPER_MOTOR_PIN1, OUTPUT);
  pinMode (STEPPER_MOTOR_PIN2, OUTPUT);
  pinMode (STEPPER_MOTOR_PIN3, OUTPUT);
  pinMode (STEPPER_MOTOR_PIN4, OUTPUT);

  pinMode (CLOCKWISE_BUTTON, INPUT_PULLUP);
  pinMode (COUNTER_CLOCKWISE_BUTTON, INPUT_PULLUP);
  pinMode (STOP_BUTTON, INPUT_PULLUP);

  groupStepper.setSpeed(10);
  Serial.begin(9600);

  stopStepping = false;
  motorDirection = 1;
}

```

در قسمت loop، بر اساس مقادیر، اتفاقات زیر رخ می‌دهد:

هر بار در ابتدای تابع، اگر موتور متوقف نشده باشد، یک step حرکت می‌کند.

```

void loop() {
  if (!stopStepping) {
    groupStepper.step(motorDirection);
  }
  ...
}

```

اگر دکمه حرکت ساعتگرد LOW باشد (فشار داده شده باشد) جهت ساعتگرد می‌شود. اگر دکمه حرکت پادساعتگرد LOW باشد (فشار داده شده باشد) جهت پادساعتگرد می‌شود. در هر دو حالت توقف موتور false خواهد بود.

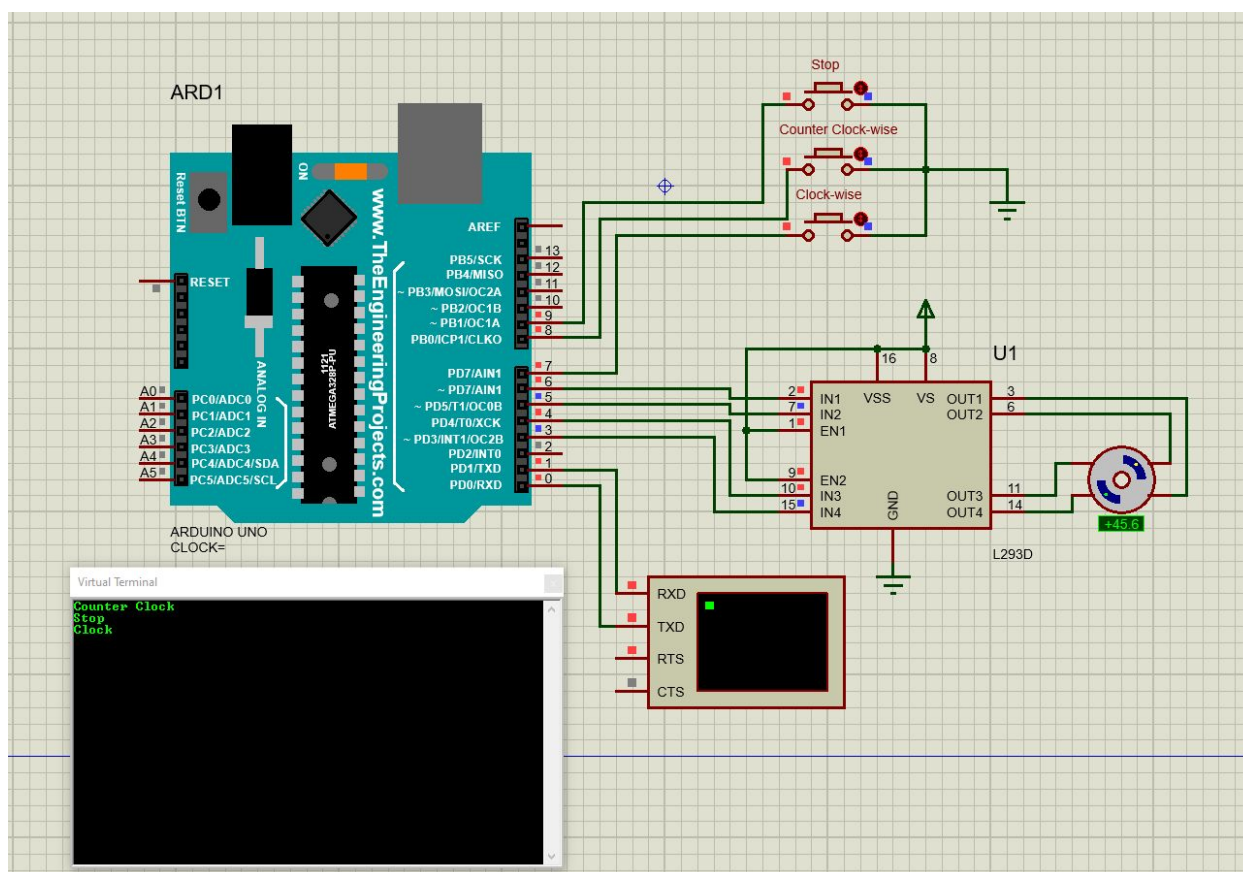
```
void loop() {  
    ...  
  
    if (digitalRead(CLOCKWISE_BUTTON) == LOW) {  
        Serial.println("Clock");  
        while(digitalRead(CLOCKWISE_BUTTON) == LOW);  
        stopStepping = false;  
        motorDirection = 1;  
    }  
  
    if (digitalRead(COUNTER_CLOCKWISE_BUTTON) == LOW) {  
        Serial.println("Counter Clock");  
        while (digitalRead(COUNTER_CLOCKWISE_BUTTON) == LOW);  
        stopStepping = false;  
        motorDirection = -1;  
    }  
    ...  
}
```

و اگر دکمه توقف LOW باشد (فشار داده شده باشد) حالت توقف موتور true خواهد بود و موتور حرکت نخواهد کرد.

```
void loop() {  
    ...  
  
    if (digitalRead(STOP_BUTTON) == LOW) {  
        Serial.println("Stop");  
        while (digitalRead(STOP_BUTTON) == LOW);  
        stopStepping = true;  
    }  
}
```

حال به بررسی شبیه‌سازی میپردازیم:

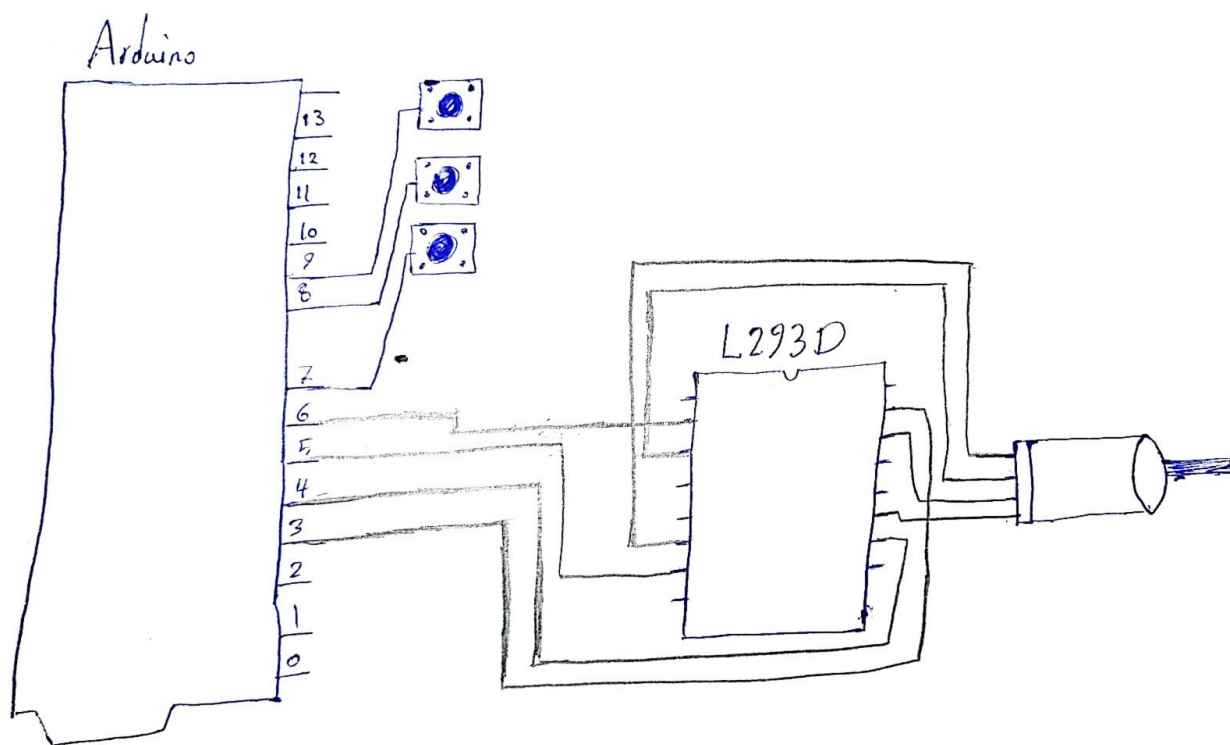
با کلیک روی دکمه‌ی Run the simulation در گوشه‌ی چپ پایین برنامه، شبیه‌سازی آغاز می‌شود و موتور به صورت ساعتگرد می‌چرخد. با زدن دکمه‌ی Counter Clock-wise جهت چرخش موتور عوض شده و با زدن دکمه‌ی Stop موتور از حرکت باز می‌ایستد. با زدن دکمه‌ی Clock-wise دوباره موتور شروع به حرکت در جهت ساعتگرد می‌کند.



نتیجه شبیه‌سازی مدار موتور Stepper




## طراحی مفهومی مدار Stepper



تصویر طراحی مفهومی مدل مدار Stepper

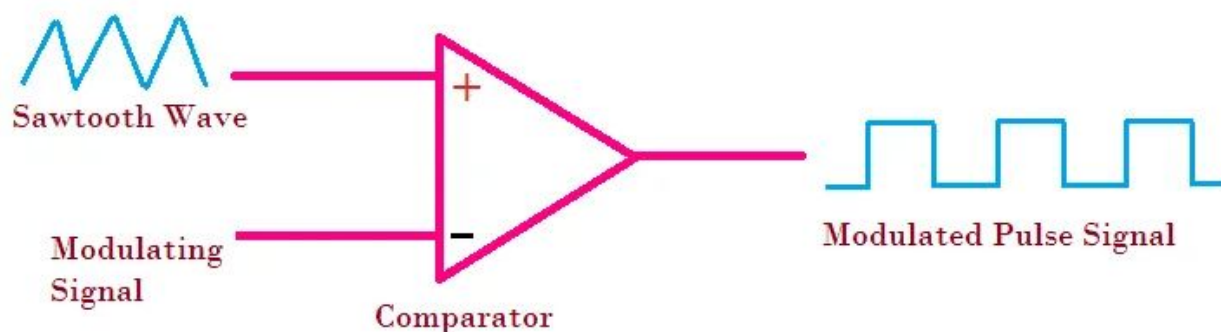
## شبه کد مدار Stepper



Pseudocode:

```
main function definition {  
  Initialization for motor state and pins;  
  while (true) {  
    if (stop_button)  
      stop the rotation of the motor;  
    if (clockwise_button)  
      start rotating the motor in clockwise direction;  
    if (counter_clockwise_button)  
      start rotating the motor in counter-clockwise direction;  
    if (motor is not stopped)  
      make an step in the motor rotation;  
  }  
}
```

## پاسخ سوال 1



با استفاده از چند موج همچون sin, ramp, saw tooth که ساخت آنها برای ما آسان است و به کمک یک مقایسه‌گر میتوان pwm را ساخت. به طور مثال در تصویر بالا یک موج دندانه اره‌ای داریم مقایسه‌گر این دو ورودی را مقایسه کرده و یک سیگنال pwm به عنوان خروجی می‌سازد، اگر مقدار موج دندانه اره‌ای بیشتر از سیگنال دیگری باشد خروجی High و در غیر این صورت خروجی Low خواهد بود.

## پاسخ سوال 2

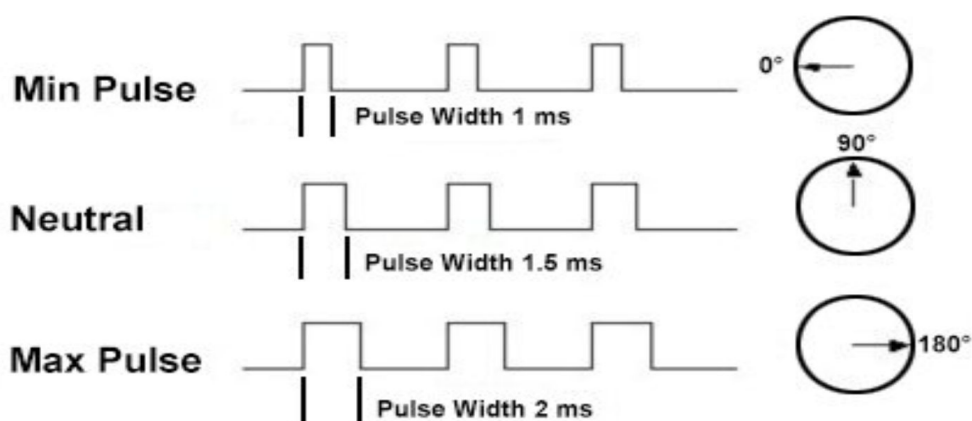
## PWM در موتور DC:

موتور DC، از دو سیم power و ground استفاده می‌کند. زمانی که به power وصل می‌شود، شروع به چرخیدن می‌کند و زمانی که از power قطع می‌شود، چرخش آن متوقف می‌شود. در هر چرخش، سرعت این چرخش غیر قابل تغییر می‌باشد و بنابراین، همواره یا با سرعت RPM در حال چرخش می‌باشد یا سرعت ۰ دارد. برای تنظیم سرعت چرخش به اندازه‌ی دلخواه (بین ۰ تا حداکثر سرعت ممکن موتور) از PWM استفاده می‌شود. در لحظه‌ی یک منطقی PWM، موتور می‌چرخد و در لحظه‌ی ۰ منطقی، موتور توقف می‌کند (این ۰ و ۱ در فرکانس بسیار بالایی اتفاق می‌افتد و کاربر یک سرعت یکنواخت را مشاهده خواهد کرد). بنابراین میزان duty cycle در pwm، مشخص کننده میزان سرعت چرخش موتور نسبت به بیشینه سرعت ممکن در آن موتور می‌باشد. برای مثال اگر از یک pwm

با duty cycle برابر با 25% و یک موتور DC با حداکثر سرعت ۲۰۰ RPM استفاده کنیم، موتور با سرعت ۵۰ RPM خواهد چرخید.

### PWM در موتور Servo:

موتور Servo، توسط PWM از طریق سیم کنترل، کنترل می‌شود. به طور معمول، هر موتور Servo می‌تواند ۹۰ درجه پادساعتگرد و ۹۰ درجه ساعتگرد نسبت به مبدأ (درجه ۹۰) حرکت کند و بنابراین دامنه حرکت آن، ۱۸۰ درجه می‌باشد. به طور معمول، فرکانس سیگنال کنترل در موتور Servo، برابر با 50Hz می‌باشد (در واقع هر ۲۰ میلی‌ثانیه، پالس مورد نظر رخ می‌دهد) و طول پالس مشخص می‌کند موتور، به چه اندازه باید بچرخد. به طور معمول، اگر طول پالس 1.5ms باشد (duty cycle = 7.5%)، موتور در حالت ۹۰ درجه (حالت خنثی) قرار می‌گیرد (تصویر وسط). در صورتی که طول پالس کمتر از 1.5ms باشد، موتور به صورت پادساعتگرد به سمت ۰ درجه حرکت می‌کند و در صورتی که طول پالس بیشتر از 1.5ms باشد، موتور به صورت ساعتگرد، به سمت ۱۸۰ درجه حرکت می‌کند. معمولاً پالس بین ۱ تا ۲ ms می‌باشد. 1ms نشان‌دهنده‌ی درجه‌ی ۰ و 2ms نشان‌دهنده‌ی درجه‌ی ۱۸۰ می‌باشد. هر زاویه‌ی دیگری بین ۰ تا ۱۸۰ را نیز می‌توان با پالسی بین ۱ تا ۲ ms، ایجاد کرد.



Variable Pulse width control servo position

### تفاوت:

بنابراین با توجه به توضیحات بالا، در موتور DC از PWM برای تعیین سرعت چرخش موتور استفاده می‌شود و در موتور Servo، از PWM برای تعیین زاویه‌ی موتور استفاده می‌شود.

### پاسخ سوال 3

#### تفاوت موتور Stepper و موتور Servo:

موتور Servo نسبت به موتور Stepper پیچیده‌تر است، سخت افزار زیادتری نیاز دارد و در نتیجه قیمت بالاتری دارد. تفاوت اصلی این دو موتور در نحوه کنترل است. موتور Stepper فقط یک عدد می‌گیرد و بر همان اساس (همان زاویه) می‌چرخد. این در حالی است که موتور Servo اینگونه نیست و باید یک فیدبک نیز از محیط بگیرد تا بفهمد که جایی که باید می‌رفته کجاست، الان کجاست و اینکه چقدر دیگر باید بچرخد.

همچنین توجه داریم که دامنه حرکت موتور Servo در بازه 0 تا 180 درجه قرار دارد در حالی که دامنه حرکت موتور Stepper بین 0 تا 360 درجه است.

تفاوت دیگر نیز به این برمی‌گردد که حرکت موتور Servo به صورت پیوسته است و هر زاویه‌ای را می‌تواند بپذیرد. این در حالی است که موتور Stepper بر اساس گام‌ها حرکت می‌کند.

موتور Servo نسبت به سایر موتورهای سرعت بالاتری دارد. موتور Stepper نیز نسبت به سایر موتورهای دقت بالاتری برخوردار است.

#### کاربردهای موتور Stepper:

- از آنجا که این موتور با استفاده از پالس ورودی به صورت دیجیتالی کنترل می‌شود، برای استفاده در سیستم‌های کنترل شده رایانه‌ای مناسب است.
- از این موتور در کنترل عددی ابزارهای ماشین استفاده می‌شوند.
- از این موتور در درایوهای نوار، درایوهای فلاپی دیسک، چاپگرها و ساعت‌های برقی استفاده می‌شود.
- از این موتور در پلاتر X-Y و رباتیک نیز استفاده می‌شود.

#### کاربردهای موتور Servo:

- درب بازکن اتوماتیک: سوپرمارکت‌ها و ورودی‌های بیمارستان نمونه‌های اصلی درب بازکن اتوماتیک هستند که توسط این موتور کنترل می‌شوند.

- موقعیت یابی آنتن: از این موتورها در محور آزمون و درایو ارتفاعی آنتن‌ها و تلسکوپ‌ها مانند آن‌هایی که رصدخانه نجوم رادیویی ملی (NRAO) از آن‌ها استفاده می‌کند، استفاده می‌شود.
- وسیله نقلیه رباتیک: این موتورها که معمولاً در کاربردهای نظامی و انفجار بمب استفاده می‌شود، چرخ‌های وسیله نقلیه رباتیک را کنترل می‌کنند و گشتاور کافی برای حرکت، توقف و راه اندازی روان خودرو و همچنین کنترل سرعت آن تولید می‌کنند.
- رباتیک: از این موتور در هر مفصل یک ربات برای تحریک حرکات استفاده می‌شود و زاویه بازوی ربات را به صورت دقیق تنظیم می‌کند.

## پاسخ سوال 4

موتور <sup>1</sup>ERM ویریه گوشی‌های موبایل، از یک موتور DC تشکیل شده که روی سر این موتور، قطعه ای با وزن نامتقارن قرار گرفته است. هنگامی که موتور DC شروع به چرخش می‌کند، به علت عدم تقارن وزن آن قطعه، لرزش ایجاد می‌شود؛ در واقع به خاطر عدم تقارن وزن این قطعه، با چرخش این قطعه موتور مدام جابجا شده و این جابجایی‌ها سبب ایجاد ویریه می‌شود. این قطعه، یک چرخش سینوسی دارد. عواملی چون نزدیکی این قطعه به میله وسط موتور<sup>2</sup> و وزن قطعه در میزان ویریه ایجاد شده دخیل هستند. در تصویر زیر، شکل بیرونی یک موتور ERM دیده می‌شود:

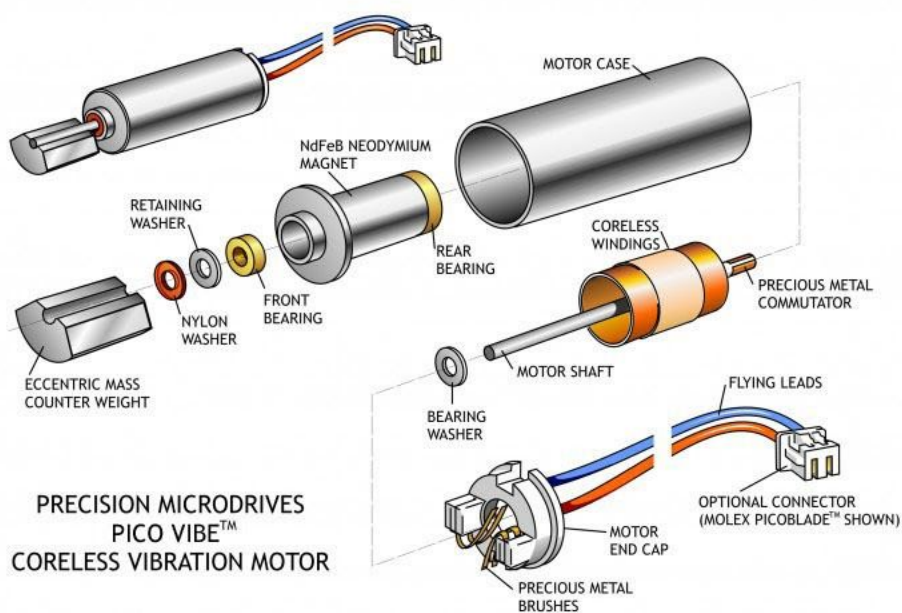
<sup>1</sup> Eccentric Rotating Mass Vibration Motors

<sup>2</sup> Shaft



شکل بیرونی یک موتور ERM

از این موتور برای ایجاد haptic feedback نیز علاوه بر وایبره استفاده می شود. در ادامه تصویری از شمای درونی یک موتور ERM دیده می شود:



شمای درونی یک موتور ERM